DIGITAL IMPLEMENTATION OF HIGH SPEED PULSE SHAPING FILTERS AND ADDRESS BASED SERIAL PERIPHERAL INTERFACE DESIGN

A Thesis
Presented to
The Academic Faculty

by

Arun Rachamadugu

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2008

DIGITAL IMPLEMENTATION OF HIGH SPEED PULSE SHAPING FILTERS AND ADDRESS BASED SERIAL PERIPHERAL INTERFACE DESIGN

Approved by:

Dr Joy Laskar, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr John D Cressler
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr David V Anderson
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: 11 November 2008

ACKNOWLEDGEMENTS

I would like to thank Dr. Joy Laskar and Dr. Stephane Pinel for their support and inspiring leadership throughout the course of this project. I would like to acknowledge Dr Bevin George Perumana for his invaluable guidance and original ideas which have greatly helped in successfully completing this work. I particularly thank him for the excellent mentor he has been and for his precious friendship. I would like to thank Dr John Cressler and Dr David Anderson for taking time and serving on my thesis reading committee. I take this opportunity to thank all the members of the Microwave Applications Group who have been directly or indirectly involved in this work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

A method to implement high-speed pulse shaping filters has been discussed. This uses a unique look up table based architecture implemented in 90nm CMOS using a standard cell based ASIC flow. This method enables the implementation of pulse shaping filters for multi giga bit per second data transmission. In this work a raised cosine FIR filter operating at 4 GHz has been designed. Various Implementation issues and solutions encountered during the synthesis and layout stages have been discussed. The design was fabricated and the fabricated chip was measured. The post silicon measurement results have been discussed and compared with the simulation results. This FIR filter design was used in the transmitter block of a 60GHz transceiver chip module.

In the second portion of this work, the design of a serial peripheral interface for initializing, calibrating and controlling various blocks in the above mentioned transceiver system has been discussed. Some modifications have been made to the standard four wire SPI protocol to enable high control speeds with lesser number of top level pads. This interface has been designed to function in the duplex mode to do both read and write operations. A unique address based SPI access has been proposed. The design has been discussed in detail along with circuit level descriptions and timing diagrams describing the circuit operation. Some ideas to improve the scope of this design have been discussed. A list of claims and innovations has been provided at the end of the document.

CHAPTER I

INTRODUCTION

1.1 Background

Transmitting signals at a high modulation rate through a band-limited channel causes Inter Symbol Interference. Pulse shaping is utilized to increase transmission data rate without increasing the bandwidth or bit error rate. It helps in minimizing sharp transitions at the input causing the frequency components of the output to remain within a specified frequency range. This enables the transmission of signals using lower transmission power. The shapes of the pulses should be chosen such that they do not interfere with each other at the optimal sampling point. To ensure this the pulse shape should decay rapidly outside the pulse interval and should exhibit zero crossing at all intervals except its own.

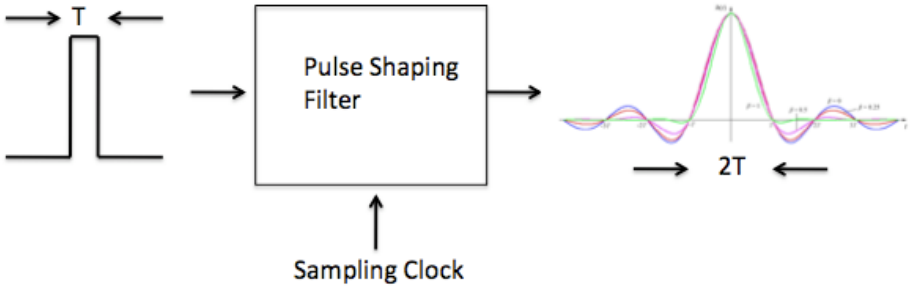

Figure 1: Conversion of an impulse to a raised cosine by filtering

The pulse shape used for transmission should have low bandwidth and also have no intersymbol interference (ISI). A sinc function has both these properties and hence can significantly increase spectral efficiency. However a system using a sinc function for pulse shaping is highly susceptible to timing jitter and phase error. This is one of the main drawbacks to be considered while designing practical pulse shaping filters. A rectangular wave filter is not as sensitive to timing jitter but needs a very high bandwidth. The raised cosine filter is commonly used in practical applications as it provides an optimal trade off

between spectral efficiency and design complexity. Shown in Figure 1 is a generic depiction of a pulse shaping filter. The impulse response of a simple pulse shaping filter is shown. The sampling clock needs to be atleast twice the frequency of the incoming data samples.

In this work, a digital FIR filter design was chosen for implementing the pulse shaping filter over an analog filter design. Digital filters can be easily integrated on silicon making them ideal for system on chip designs. FIR filters were chosen over IIR digital filters as they are more stable and can be designed to have linear phase and require no feedback.

Mathematically digital filters can be expressed as a convolution of the input samples and the impulse response of the filter.

$$y(n) = x(n) * h(n) \tag{1}$$

This can easily be implemented in the digital domain using a set of shifters, multipliers and adders since the above equation can be represented as :

$$y(n) = \sum_{i=0}^{T-1} x(n-i)h(i) \tag{2}$$

where,

x(n) is the input data stream

h(n) is the impulse response of the filter

T is the number of Taps in the filter

The number of taps chosen for the filter determine the nature of the output frequency response. A large number of taps determines the duration of the impulse response. The trade off between increasing the number of taps which increases design complexity and the circuit's output response is a design choice. Another important aspect of digital filters is the amount of oversampling. It turns out that the filter must oversample the data by a factor of atleast 2 (Nyquist rate) to obtain a useful response characteristic.

1.2 Motivation

With the advent of multimedia streaming and other high data throughput applications, the speed of operation for pulse shaping filters is higher than ever before. Conventional design

implementations are no longer sufficient to meet these requirements. Traditional FIR filters are implemented as a series of multipliers, adders and delay lines. As the number of taps increases, the design of the filter for a high frequency becomes rather involved. The adder design is not a trivial problem at higher frequencies.

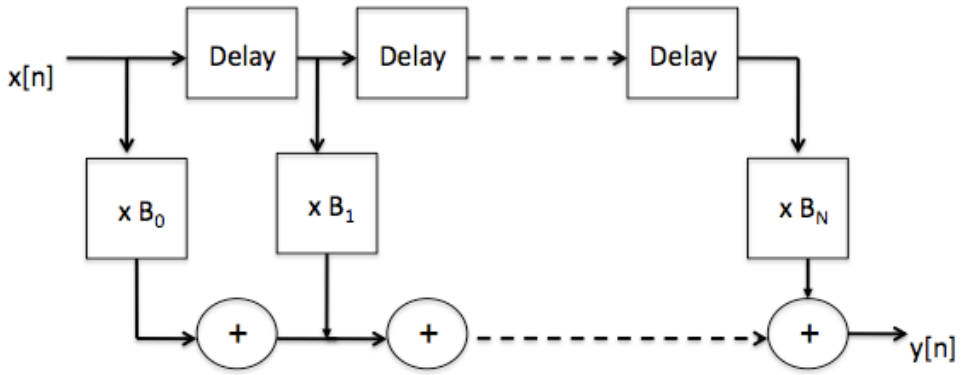

Figure 2: Block diagram of a standard FIR filter using multipliers and adders

As we can see in Figure 2 the carry propagation for high speed adders is a design challenge in itself. To obtain the final output of the system, one must wait for the result of the last addition operation.

Consider a 13 tap filter for a 2Gbps wireless transmission. With an upsampling factor of 2 the filter has to operate at 4 GHz.

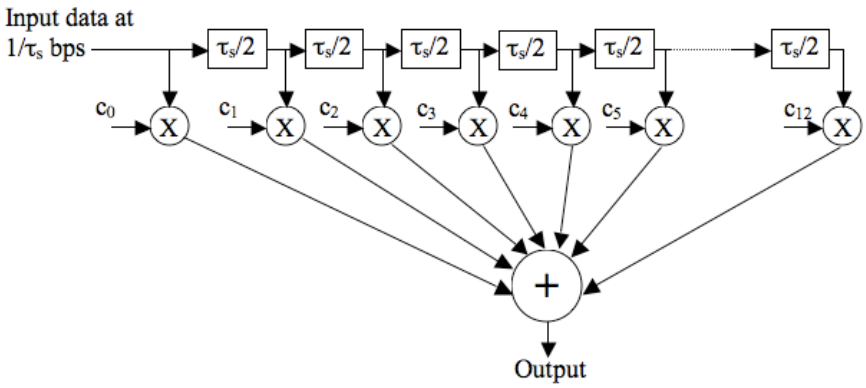

Figure 3: Block diagram of a traditional 13 Tap Filter with an Upsampling factor of 2

Shown in Figure 3 is a functional block diagram of the traditional implementation of the filter in this configuration. Since the number of taps is 13, from Equation 2 it can be

seen that it needs 13 addition operations. The addition from the last stage needs to wait for the carry to propagate from the first addition stage.

At such high frequencies the implementation shown in Figure 3 can prove to be a bottle neck as it requires the implementation of really high speed adders and multipliers. In order to simplify this design and enable the operation of pulse shaping at higher frequencies a unique look up table based approach has been proposed.

CHAPTER II

LOOKUP TABLE BASED IMPLEMENTATION

2.1 Concept

The lookup table approach involves realizing the filter directly from its truth table. This approach is effective because it can be observed that many coefficients in the filter expressions are actually zero terms. In the traditional implementation these zero terms would have still constituted a MAC (multiplier and add stage) stage. When a lookup table was constructed and the logic realized it was noticed that an optimized logic block could be inferred. Modern synthesis tools like Build Gates and Design Compiler come with very powerful logic optimization algorithms that can infer a truth table and convert that to its simplest form. In this approach, the coefficients are divided into odd and even sets. The logic for each of these sets is realized separately and the outputs are muxed together with the clock as select line. Having N parallel paths and then performing a parallel to serial conversion at the end can achieve upsampling by a factor of N.

2.2 Example

To better understand this approach let us look at an example. This technique is used in implementing a 13-tap raised cosine filter with a roll-off factor of 0.25.The tap coefficients in this implementation are:

$[0.0000, 0.0866, 0.0000, -0.1856, 0.0000, 0.6274, 1.0000, 0.6274, 0.0000, -0.1856, 0.0000, 0.0866, 0.0000]$

Since the upsampling factor is 2, the coefficients are divided into two sets :

$Odd : [0.0000(d0), 0.0000(d1), 0.0000(d2), 1.0000(d3), 0.0000(d4), 0.0000(d5), 0.0000(d6)]$

$Even : [0.0866(d0), -0.1856(d1), 0.6274(d2), 0.6274(d3), -0.1856(d4), 0.0866(d5)]$

Since there is only one non-zero coefficient in the odd set, the output of this path is just d3, which is the input data delayed by 3 time periods. However, the output of the even path can be any one of the 2^6 possible outcomes since there are 6 non-zero coefficients.

For example if the last bits were all 1s, the output would be 0.0866-0.1856+0.6274+0.6274-0.1856+0.0866 = 1.0568 and so on. In order to run at extremely high speeds, all the possible 2^6 outcomes can be precalculated and stored at addresses representing the last 6 data inputs. Table 1 lists all the possible outcomes as 6 bit binary numbers (obtained after scaling and adding a bias). Also shown at the bottom of Table 1, are the 2 possible outcomes of the odd path.

Table 1: Precalculated 6 bit outcomes for all possible 6 bit data inputs

Address	Outcome	
000000	010101	*Even Coefficients Outcomes*
000001	010111	
000010	010000	
⋮	⋮	
111101	111010	
111110	110011	
111111	110101	
Odd (d3=0)	010101	*Odd Coefficients Outcomes*
Odd (d3=1)	110100	

The digital delay chain implementing the 6-bit address is shown in Figure 4. Each of the precalculated outcomes in the even or odd path can be stored as a 6-bit word in memory. However, the use of memory can be avoided if we determine each bit outcome as an output of a combinational logic block with 6 inputs (the address bits).

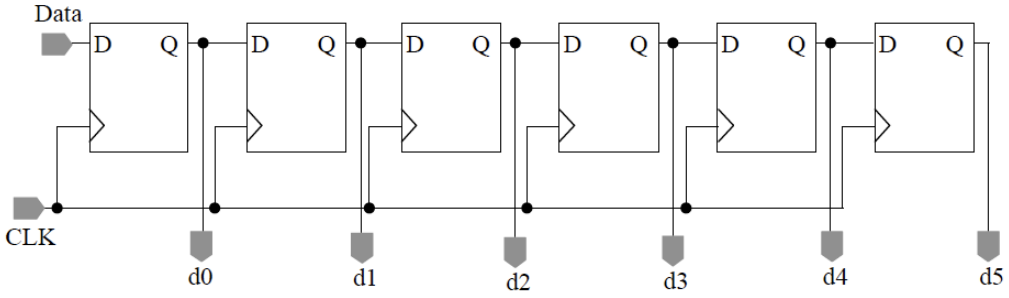

Figure 4: Digital Delay line for generating a 6 bit address

This implementation is shown in Figure 5. Each of the 6 bits in the output word is evaluated as an output of the even or odd combinational block. Six multiplexers choose the results of the odd path in the positive half of the clock cycle and the even path in the negative half of the clock cycle. This parallel to serial conversion implements the upsampling. The 6-bit output word is fed to a digital-to-analog converter that generates the analog output. The even and odd paths have D flip-flops at the end so as to latch the output. The clock applied to these flip-flops is delayed to compensate for the delay in the combinational blocks. The clock fed to the multiplexer select terminal is further delayed to compensate for the clock to output delay of the flip-flops at the output of odd and even paths. The disadvantage of this system is that as the tap count of the filter increases, the complexity involved in realizing the look up table also increases.

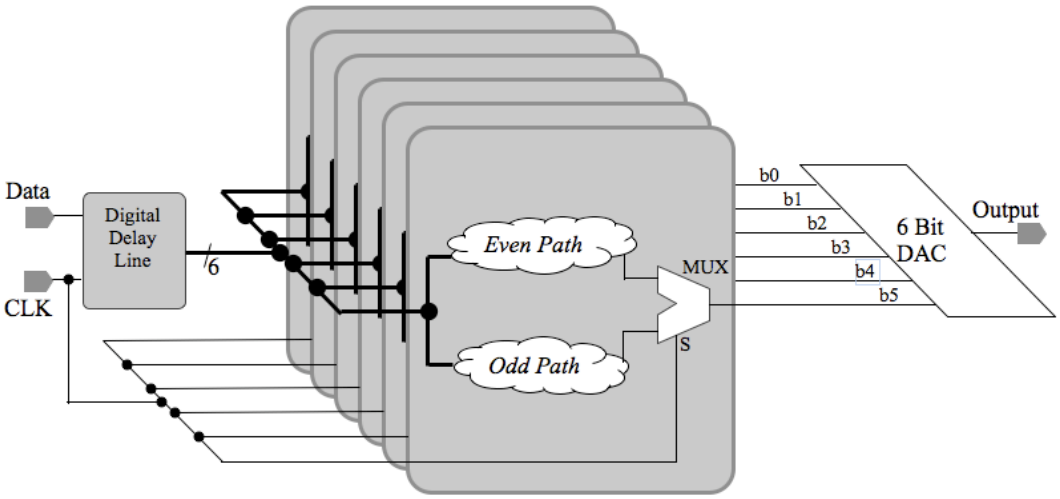

Figure 5: Implementation of the 13-tap raised cosine pulse shaping filter as a look-up table with precalculated outcomes in even and odd paths

CHAPTER III

ASIC IMPLEMENTATION

In this chapter the specific details of all the steps involved in the design, from conception to fabrication are discussed.

3.1 MATLAB Implementation

The filter coefficients were generated using MATLAB. A matlab code was written to process all the input combinations from Table 1 and compute the lookup table based on Equation 2. This was quantized and encoded into a binary bit sequence and approximated so as to minimize the quantization error. The variation of changing the resolution of the output was studied and an optimal choice of the resolution was decided. It was also observed that there was some clock feedthrough if the input levels were chosen as 0 and 1. This resulted in a spike in the output spectrum as shown in Figure 6. When the levels were changed to +0.5 and -0.5 the frequency spectrum is not distorted. The correction factor of 0.5 was chosen after a range of values were evaluated. Despite the correction factor, some clock feedthrough was observed in the simulations (possibly due to stray capacitive coupling effects from neighbouring routes in the layout).

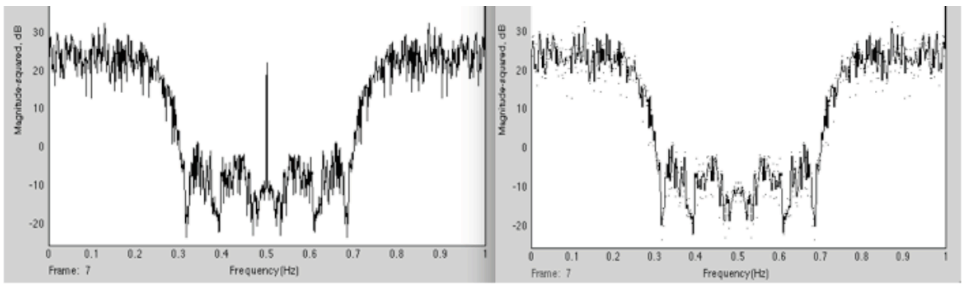

Figure 6: Matlab generated frequency spectrum plots with and without the correction factor

A simulation setup was created using simulink to model the lookup table based filter and the output was verified. Shown in Figure 7 is a snapshot of the lookup system modeled

in matlab.

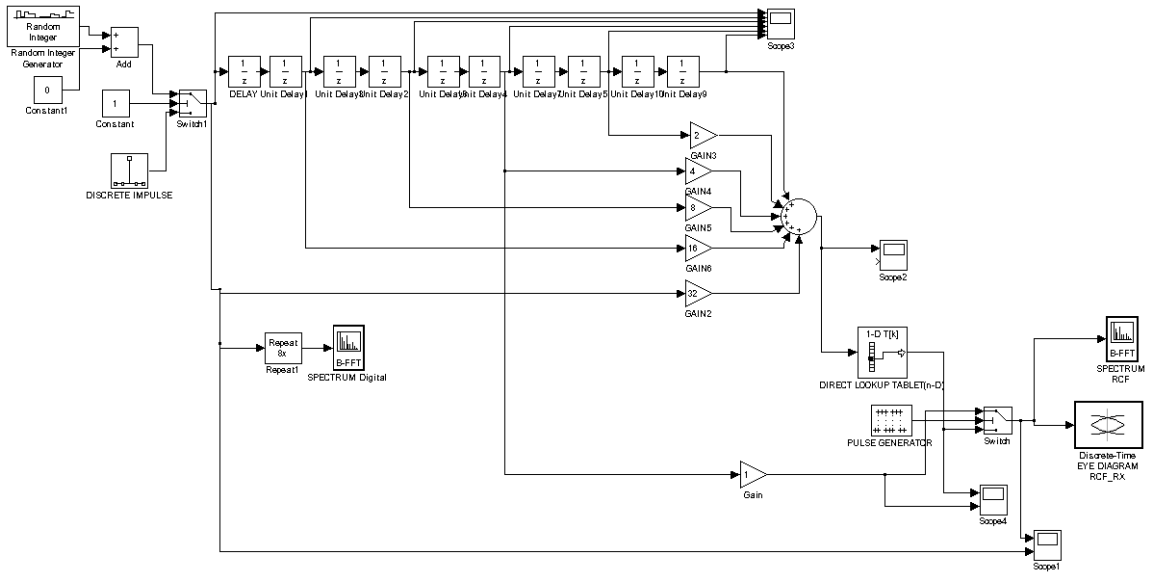

Figure 7: Simulink model implementing a 13 tap 6 bit LUT based FIR filter

3.2 RTL Generation

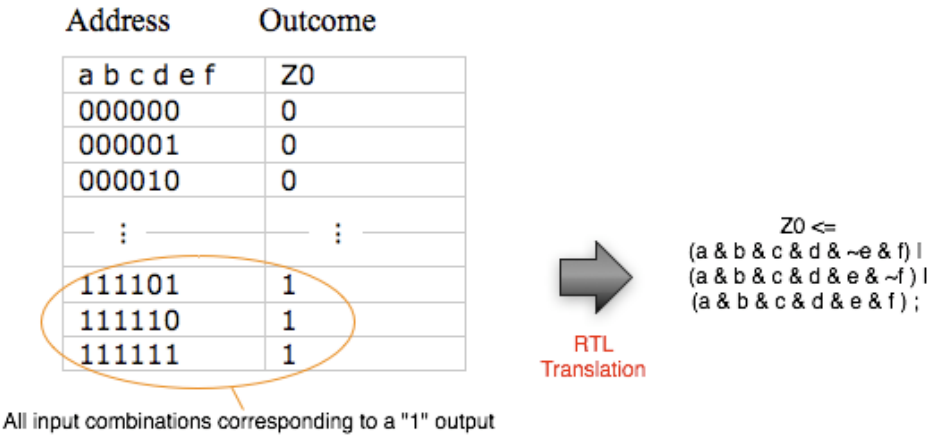

Figure 8: Example of RTL translation for the look up table

Once the lookup table is created from Matlab, this is converted to RTL. Scripts were written in Perl to automate this process. The lookup function is realized as a sum-of-products form i.e all those combinations of inputs corresponding to a 1 in the output are considered and realized as a logic expression.

For example, from Table 1 if only one of the outputs is observed and translated to an equivalent verilog RTL construct, it would look like the translation described in Figure 8.

Once the RTL is ready, we are ready to take the design through the back end design flow. Shown in Figure 9 is an overview of the backend flow. The flowchart also shows the information needed by the user from the foundry at different stages of the design flow. The tools used in this work are Build gates, SOC Encounter and ICFB from Cadence.

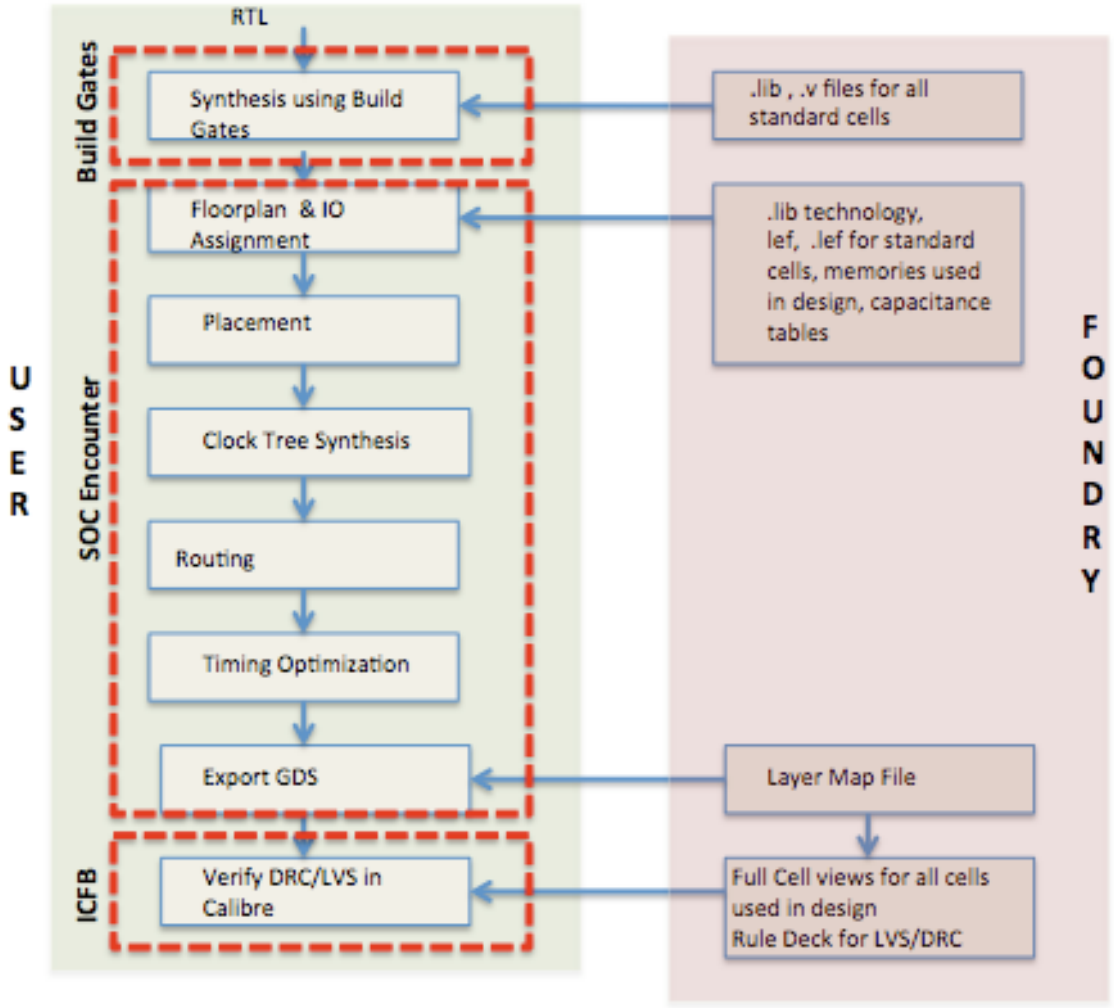

Figure 9: Typical ASIC flow

3.3 Build Gates For Synthesis

In this stage, the RTL is analyzed, synthesized and optimized. The look up table is realized as a set of sequential paths from input registers to output registers. The even and odd

paths in the logic have been realized separately and multiplexed together in the layout stage. The tool used for synthesis is Build Gates from Cadence. The synthesis process begins by converting the verilog description of the logic into a netlist using generic library independent gates. These are targeted to specific standard cell libraries depending on the speed and power requirements. In this work, standard cell libraries from STMicroelectronics have been used. Standard cell libraries are characterized based on functionality, power and performance. Libraries using low threshold voltage cells can operate at higher speeds but come with an associated higher leakage power consumption penalty. In order to estimate area and perform static timing analysis, timing models in liberty format need to be provided. The tool uses these timing models in worst, best and nominal PVT corners to estimate the early and late slack in the design. There is no routing information at this stage. The delay from the interconnects is modeled using wireload models which need to be provided to the tool. These are statistical models that offer an approximate estimate of the delays between cells. As we can see delay estimates based on wireloads are approximate ballpark numbers and should not be considered as an accurate reference. The next important input to the synthesis tool is the timing constraint file. In this work, the timing constraints have been described in the Synopsys Design Constraint format. Here the clocks are specified. The latencies of the clock, the false path declarations, the input and output delays are provided as inputs. Since the filter design is to be operated at high frequencies, it is required to target the optimization for timing. Logic restructuring, remapping cells for timing are some of the functions of the synthesis tool. Before proceeding to the layout stage it is critical to clear up the setup violations. Hold time violations can be corrected in the layout stage as interconnects and metal routes will add to the delay helping solve these issues. But significant setup violations involve datapath restructuring which is optimal if done at the synthesis stage to avoid iterations in the back end design stage. Once the synthesis is complete the post synthesized verilog netlist is generated.

3.4 Strategy for timing closure during Static Timing Analysis

Static timing analysis is a method of computing the expected timing of a circuit without simulating it. This uses characterized timing models and delay tables to estimate the data and clock arrival times. The two important checks done by the STA engine of a tool are setup and hold checks to avoid the flops of the design from entering a metastable condition. To better explain this concept consider the following synchronous data path:

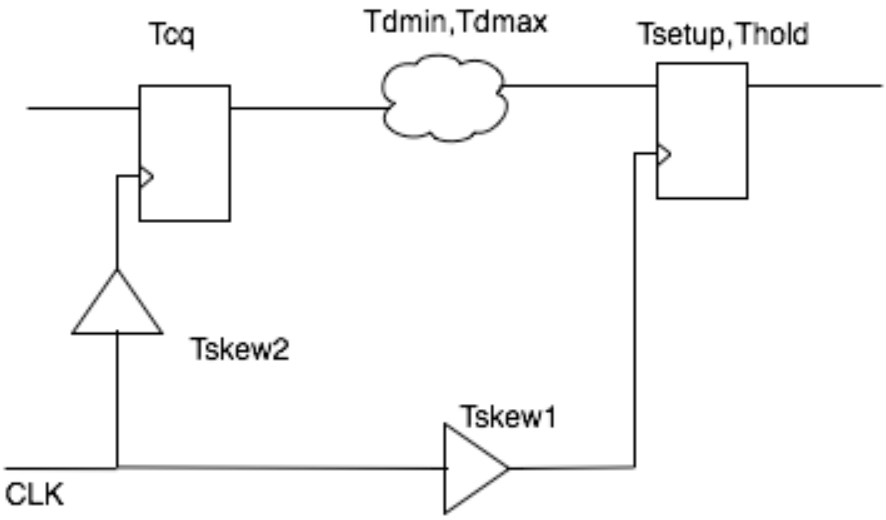

Figure 10: Setup and Hold Analysis

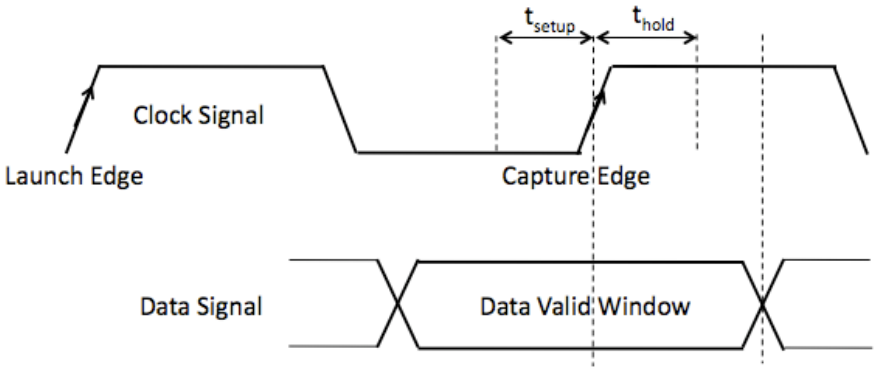

Figure 11: Setup and Hold Check Timing waveforms

- Setup Time : The time for which the data has to be stable before the clock edge

arrives

- Hold Time: The time for which the data has to be stable after the clock edge arrives

For the Figure 10 to avoid a setup violation,

$$T_{clkmax} + T_{skew1} - T_{setup} > T_{cq} + T_{dmax} + T_{skew2} \tag{3}$$

To avoid a race condition due to a hold violation,

$$T_{hold} + T_{skew1} < T_{dmin} + T_{skew2} + Tcq \tag{4}$$

where,

T_{cq} is the clock to Q delay of the flop

T_{clkmax} is the maximum time period of the clock

T_{skew1}, T_{skew2} are delays in the clock network

T_{setup}, T_{hold} are the setup and hold times for the flop

T_{dmax}, T_{dmin} are the maximum and minimum datapath delays

Since this is a high speed design, the entire system is implemented as a sequential path from input to output. For each path, the datapath was optimized to meet the setup requirement shown in the Equation 3. However it was observed that, this still resulted in some negative slack. On evaluating the setup margin and hold margin it was observed that on manually inserting delay in the clock path by buffering the clock network without causing a negative hold slack, the setup violation was resolved. Inserting artificial skew is sometimes not recommended in large synchronous systems which share the same clock as this can cause unpredictable clocking operation and add to jitter. In this case, since this is the first stage of the transceiver lineup, the clock is tapped from the filter and acts as a source clock to the rest of the system. The delay inserted manually therefore only contributes to the source latency in the clock path and can be ignored if it lies within the latency specifications of the system. As we can see, trying to improve the system timing to resolve a setup violation using clock skew can eventually cause a hold violation. Setup violations are clock dependent. If the clock frequency is reduced the setup violations can

vanish but hold violations are more risky to have as they are frequency independent. They are functions of the minimum datapath delay, clock skew and hold time of the library cells. One can live with setup violations at the end of a design cycle, but hold violations cause race conditions that can kill the system as they cannot be corrected by reducing the clock frequency. Another technique that was implemented for meeting STA was to target all the standard cells used to a lower threshold voltage library. Low threshold standard cells can operate at much higher frequencies but come with an associated higher leakage power penalty.

3.5 SOC Encounter for Automatic Place and Route

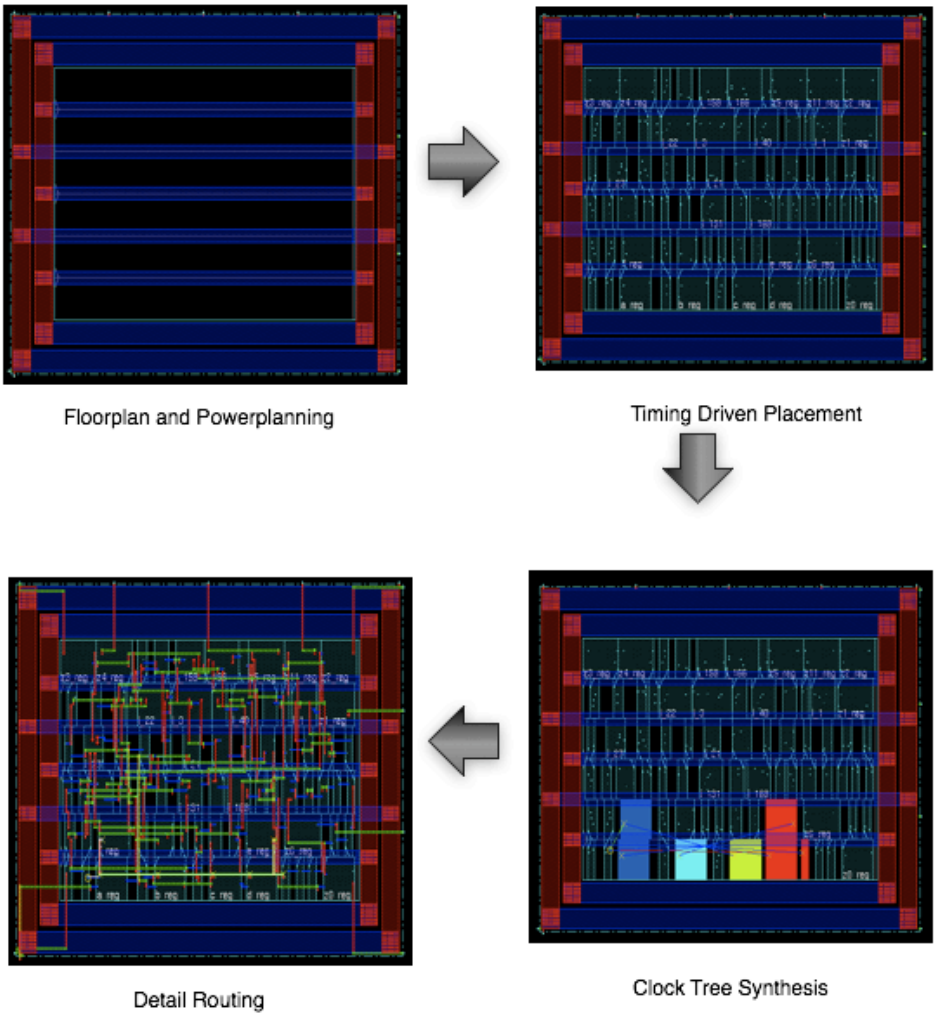

Figure 12: Important stages in a typical backend flow

The post synthesized layout is now ready for back end layout design. The tool used for automatic place and route is SOC Encounter from Cadence. Along with the netlist, additional timing libraries and physical libraries need to be provided. The standard library cells need to be provided in Library Exchange format. This has all the physical properties of the cells. The locations of the pins of the cell, the metal layers it uses and the actual physical dimensions of the sub block. This along with liberty timing models provides a good idea to the tool on the design considerations for the layout. There are specific design rules that are stipulated by the foundry and need to be strictly adhered to for reliable manufacturing. These design rules are usually provided to the tool in the form of a technology LEF file. This usually contains information on via and metal specifications, spacing constraints, pin guidelines etc.

3.5.1 Floorplan and Powerplanning

The first stage of the Encounter flow is floorplanning. Here the core boundary is decided. This can be provided as an aspect ratio or as explicit chip dimensions. The top level IO pin assignments happen at this stage. The location of pins are identified and the metal layers for each pin are chosen. Following IO assignment, power routes are laid. The power routing is done before the signal routing to ensure that a well distributed power ground mesh is laid out before the standard cells are placed. This is needed to reduce the static IR drop in the system. The ground lines are chosen to be in metal 1 since they can directly connect to the ground mesh fabric of the chip. This ensures a well distributed ground plane reducing effects such as ground bounce. The power mesh is created as a wide metal ring around the chip. Metal strips run across the mesh to provide connectivity to intermediate cells. Once the power routes are laid at the top level, standard cell power lines are routed. These are metal lines connecting the power lines of each standard cell to the outer power mesh. In order to save area, the floorplan in this design has been done using double back rows where every alternate cell row is flipped so that the vdd and gnd for two consecutive rows of standard cells are shared.

3.5.2 Placement

The next step is to run a timing driven placement of the standard cells in the design. This ensures that constraints in the sdc file are met at the placement stage. This is done by placing the cells based on their flyline connectivity to the top level IO pins. Flylines are imaginary lines indicating the connections of a given cell to its neighbors. The length of the flyline needs to be as small as possible to reduce interconnect lengths of the routes. If there are issues in the setup analysis after placement, timing optimization can be run to improve the datapath delay.

3.5.3 Clock Tree Synthesis

Once placement is completed, the next stage is clock tree synthesis. In large designs, it is critical to lay the clock and high fanout nets such as reset before laying the regular routes. If clock nets are routed along with the signal nets the tool would consider routing it like any other signal net. This may cause the insertion of a degraded unbalanced clock tree. A clock tree is built at this stage. This again is done as per certain clock tree specifications of skew and tree depth. The tool buffers the clock tree to meet the loading requirements based on the number of sinks the clock port sees. As discussed before, if the skew is not within acceptable limits, it can cause unpredictable behavior.

3.5.4 Routing and Optimization

After clock tree insertion, the next step involves routing the design. Routing happens in two stages: Global routing and Detail routing. Global routing is a quick routing scheme where the layout is divided into blocks and the congestion in each block is analyzed. This is done before detailed routing to get an estimate of the congestion in the design. Detailed Routing is the stage where metal lines are assigned to the signal connections. Post route timing includes delays calculated from the interconnects as well. This information can be written to an SDF (Synopsys Delay Format) file. The gds can be extracted from this layout. After routing, static timing analysis is done to check if there are any setup and hold violations. All setup and hold violations need to be fixed at this stage. Violations that are not resolved

by the tool need to be verified and corrected manually. Hold violations can be solved by inserting delay elements in the datapath or reducing the delay in the clock network. The former method is preferable if there is sufficient setup margin. Setup violations need to be solved on a case by case basis. Standard cells in the datapath can be resized, data path logic can be restructured to reduce the delay. Whenever any element is added to the layout which was not present in the netlist, it has to be done through an ECO (Engineering Change order) operation. Using the timing debug ECO flow in Encounter, cells can be easily added on a net. This should be followed by an ECO place and an ECO Route to modify only the necessary routes without changing the other routes. In order to run an extracted simulation it is necessary to generate a SPEF (Synopsys parasitic extraction format) file. Once the routing is complete, it is required to add filler cells. Filler cells are standard cells which mainly have the well layers. These come in varying widths and are used to fill the gaps in the design. They ensure N Well continuity. A discontinuous NWell is a DRC violation and needs to be corrected. During timing optimization, the tool adds cells and replaces some cells with those of higher/lower drive strength. This essentially modifies the original netlist. This modified netlist needs to be dumped from the tool and saved for later use. Shown in Figure 12 is a broad outline of the important steps discussed.

3.6 Top Level Layout Integration

The gds of the FIR filter core is now streamed into ICFB. Here the top level test structure is designed. The output of the filter core drives a DAC. The clock and data lines are fed to the core block through top level input pads. The analog output from the DAC goes through a buffer to be able to drive the output pad. Here decoupling capacitors are added between the vdd and gnd to reduce any dynamic IR drop and ground bounce. All the open spaces in the layout are filled with a uniform repeating ground fabric to reduce the ground resistance. The buffer connected to the output pad is designed to drive 50ohms. Diodes were used to resolve antenna violations which occur in long stretches of metal connected to a gate. To reduce substrate noise and hold the substrate at the ground potential, a sufficient number of PTAP substrate contacts are provided throughout the layout. At this

stage the design is ready for LVS and DRC checks. Calibre from Mentor Graphics was used to run a design rule check. DRC check ensures the correctness of physical spacing, width and via requirements for the layout. It also checks for density violations. LVS check ensures equivalence between the layout and the post layout netlist. The top level layout sent for fabrication is shown in Figure 13. The DAC used in this layout is an R2R 6 bit DAC that is capable of operating upto 6 Ghz. Care must be taken while connecting the FIR to the DAC. The lines connecting the DAC and FIR need to be done carefully to ensure equal length routing. This is to avoid the DAC to compute a wrong state due to a mismatch in the arrival times of the individual output bits from the FIR due to differing interconnect lengths.

3.7 Simulation and Functional Verification

The post layout netlist was read into Spectre Simulator and the functionality was verified using a transient simulation. On computing the discrete fourier transform of the transient signal, the spectrum was analyzed and the suppression at the designed frequency was observed. The specifications of the filter design are shown in Table 2. Post Layout extracted simulation can be run by using the extracted view of the layout from Cadence and running the same transient simulation as described above.

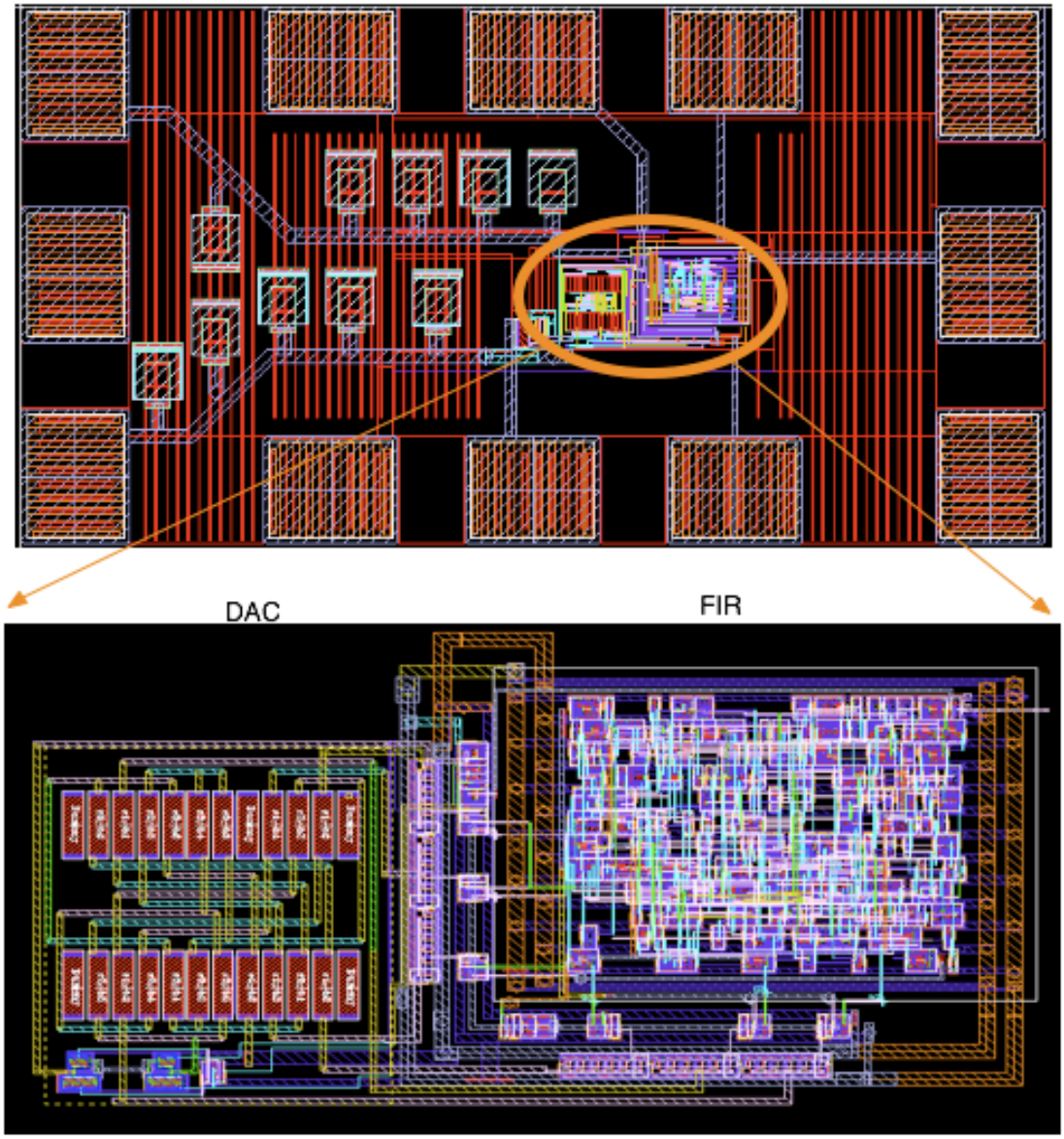

Figure 13: Full Layout of FIR test structure

CHAPTER IV

MEASUREMENT RESULTS

The fabricated chip was externally probed and a spectrum analyzer was used to study the output of the DAC. The figures below illustrate the performance of the design.

4.1 Specifications of Filter

Table 2: Specifications of Filter

No of Taps	13
Roll Off Factor	0.25
Suppression	28dB
Upsampling Factor	2
Clock Frequency	Works upto 4Ghz
Vdd	1V
Process	90nm CMOS
Filter description	Raised Cosine FIR

4.2 Eye Diagrams

The Eye diagrams show that there is an opening all the way upto 3.8 GHz (using 1V Vdd). With a 1.1V Vdd the eye diagram looks good all the way till 4.3GHz. At lower frequencies, the eye diagram appeared degraded since the dc block that was used was not designed for very low frequencies.

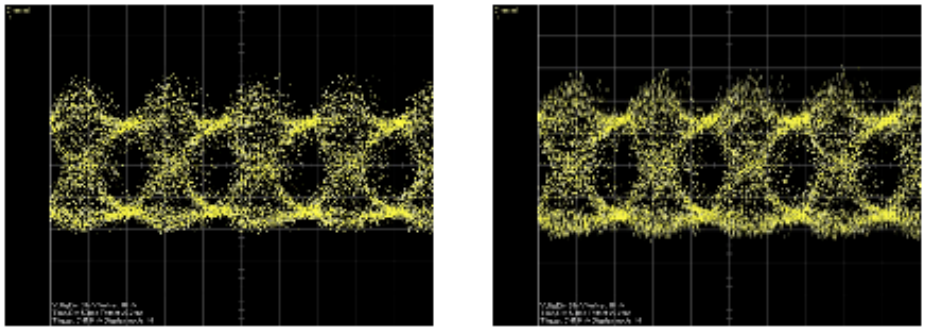

Figure 14: Eye Diagram at 864Mbps with Vdd=1V and Vdd=1.1V

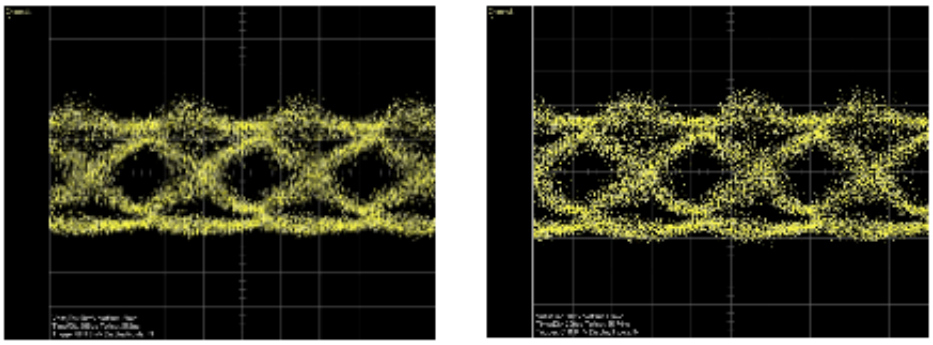

Figure 15: Eye Diagram at 1728Mbps with Vdd=1V and Vdd=1.1V

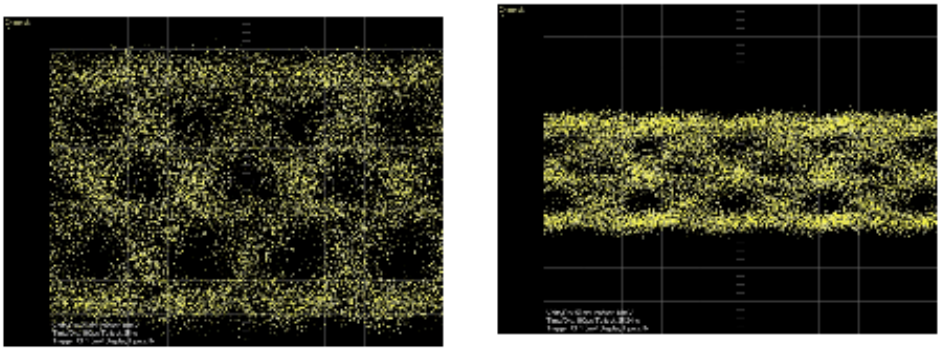

Figure 16: Eye Diagram at 3.8Mbps with Vdd=1V and Vdd=1.1V

4.3 Frequency Spectrum and Impulse Response Plots

The frequency plots at various data rates are shown. Though the suppression is as expected, there is some clock leakage in the output plot. At lower frequencies, the clock leakage is almost absent. This observation suggests that the clock leakage is a function of the layout (possibly some stray clock coupling from a neighboring line). On connecting multiple FIRs in a differential configuration the clock leakage almost vanishes. Figures 17 - 20 show the frequency spectrum plots for a wide range of data rates.

At lower frequencies, the impulse response can easily be identified as seen in Figure 21. At higher frequencies, the charging and discharging of the capacitive nodes makes the impulse response almost look like a smoothened curve as seen in Figure 22. The impulse response was measured by feeding an input bit stream of a string of 0s and a single 1. The expected response is the coefficients of the filter which can be observed in the plots below.

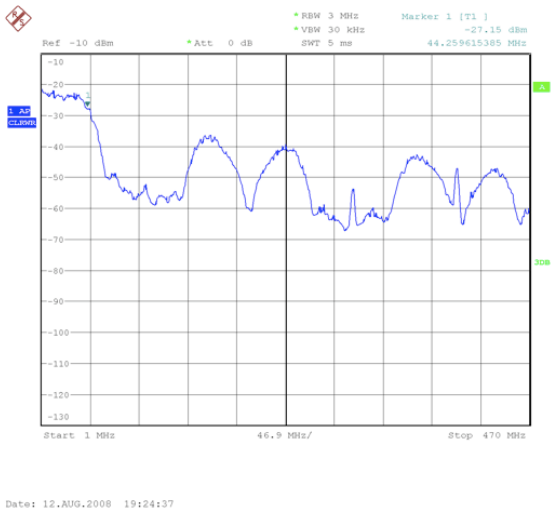

Figure 17: Frequency Spectrum at 100 Mbps

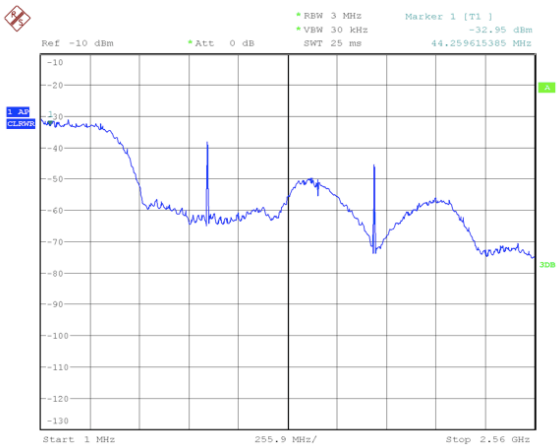

Figure 18: Frequency Spectrum at 864 Mbps

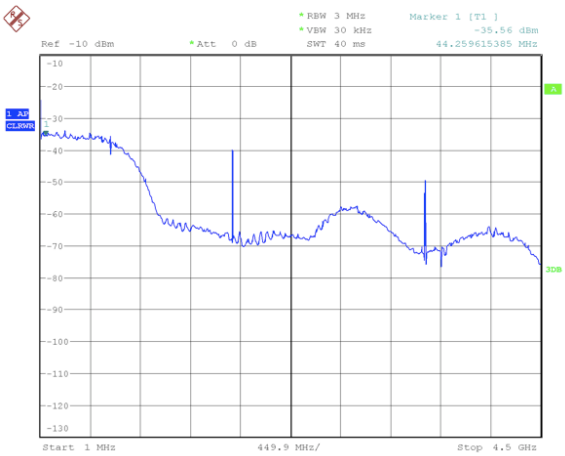

Figure 19: Frequency Spectrum at 1728 Mbps

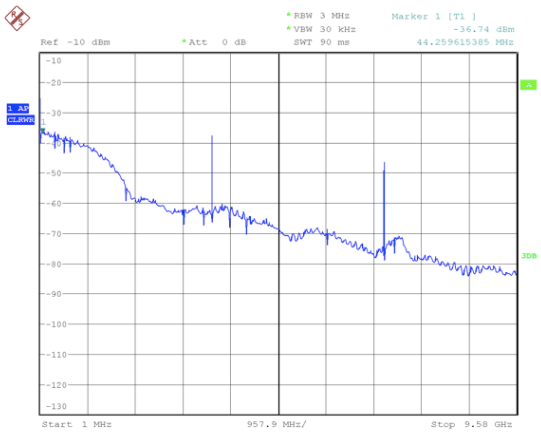

Figure 20: Frequency Spectrum at 3456 Mbps

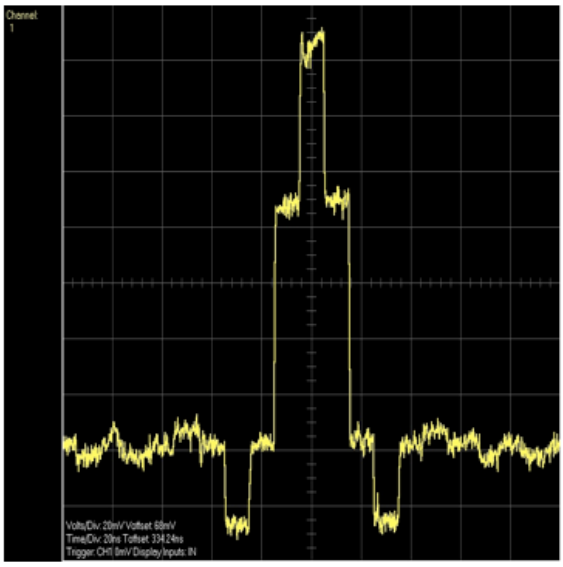

Figure 21: Impulse Response at 50 Mbps

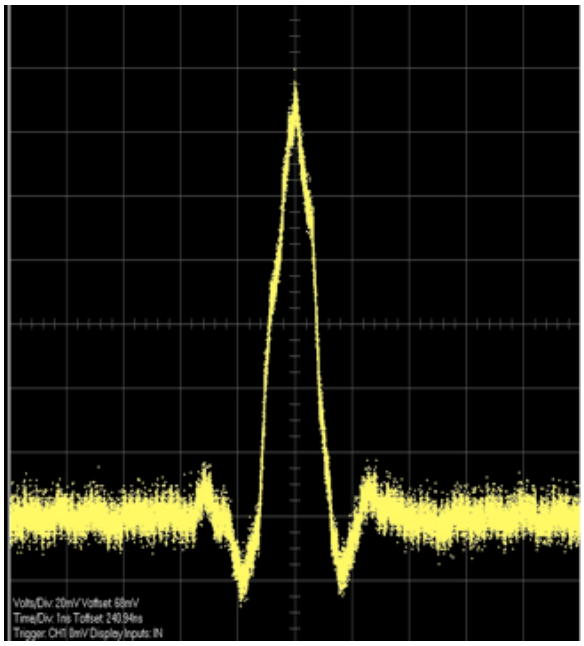

Figure 22: Impulse Response at 1728 Mbps

CHAPTER V

ADDRESS BASED SERIAL PERIPHERAL INTERFACE

5.1 Background

In a transceiver system, there are many modules which need initializing, monitoring and calibrating. For eg, before switching the transceiver from transmit mode to receive mode a number of control bits need to be asserted to change the states of some internal nodes in the logic. If this were to be done in a traditional way, it would require many control inputs in the top level. Today, designs are strictly pad limited. To overcome this limitation, serial interfaces such as the SPI are used. The traditional SPI topologies are described in Figure 23.

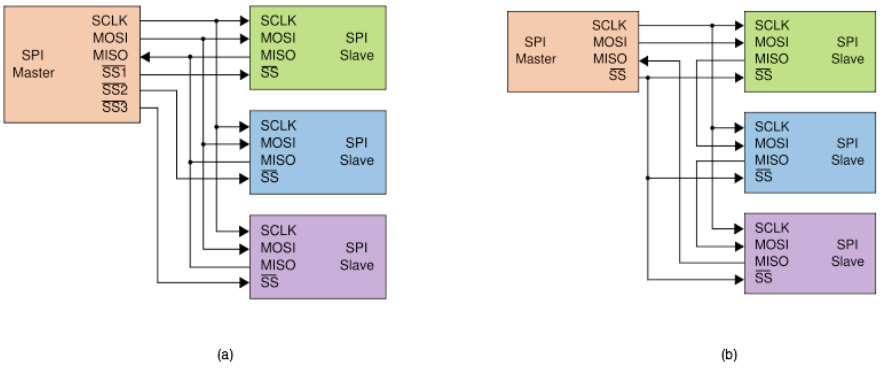

Figure 23: Traditional SPI topology with (a) multiple slave select ports and (b) Daisy Chain Topology (source: Wikipedia[11])

The disadvantage of the traditional topology in Figure 23 (a) is that it needs multiple slave select ports. A modification to this approach is the daisy chain topology shown in Figure 23 (b). This needs only one slave select port but it is much slower in operation because of the serial interconnection of the slaves. In order to access the last SPI one must wait for the data word to propagate from the first slave to the last slave. In higher end transceiver systems, the initialization needs to happen in short intervals of time. To overcome the limitations of speed and pad count in traditional topologies the address based

topology is proposed.

5.2 Address based SPI Topology

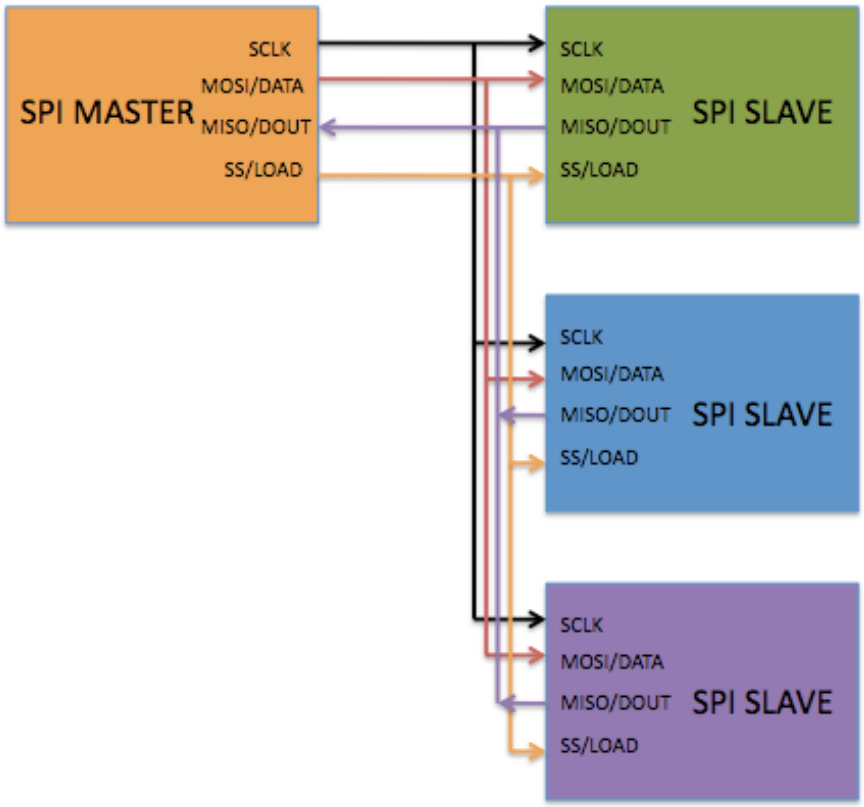

Figure 24: Address Based SPI Topology

In the address based SPI topology every slave in the SPI system is assigned a unique address. Similar to a memory lookup, in order to access a particular SPI slave, it needs to be addressed first. The data and clock lines are connected to all the slaves. The data to be written to the SPI is prepended with a control sequence which constitutes the address of the target slave and some control bits to enable read or write mode. As it can be seen in Figure 25, in write mode the SPI functions as a serial to parallel converter and in the read mode, it functions as a parallel to serial converter.

The address check block compares and verifies if the input data sequence is for the current SPI by comparing its hardwired address with the address in the received sequence in the shift register. The Read Block performs parallel to serial conversion where the data

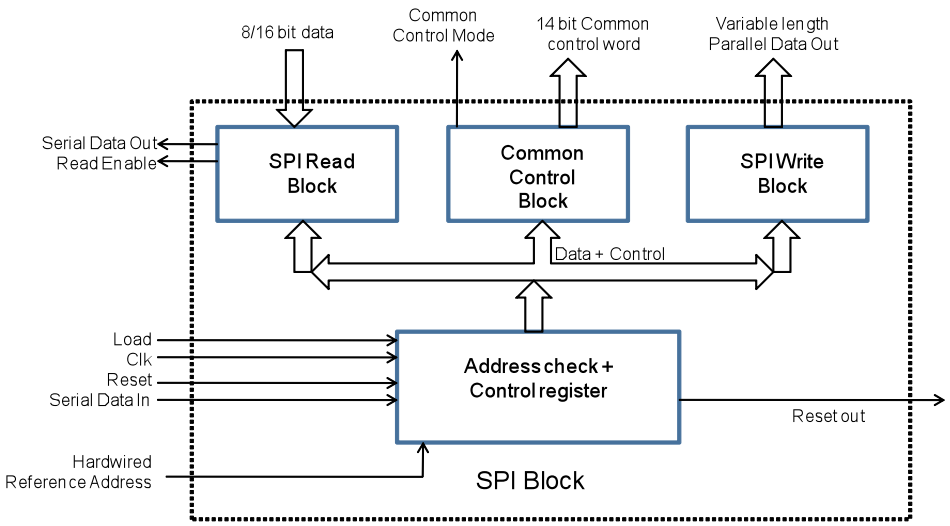

Figure 25: Block Diagram of Top Level Functions of the Address Based SPI Implementation

bus from the core logic is transferred to the Serial Data Out Line. The SPI Write Block performs serial to parallel conversion where the serial data stream in transferred to the core logic if the address matches. The common control block is a mode useful during initialization where a common word address is used to transfer data. This common word address is shared by all SPI slaves and in the common mode all slaves are loaded with the same data sequence.

5.3 Circuit Design and Description

A slightly more detailed description of the circuit used to implement this system is shown in Figure 26. This shows a few of the important functional modules of the system.

The load signal is a critical signal in this setup. The load is the control signal which issues a transfer signal for the data from the SPI registers to the slave registers. Depending on the length of the shift register in the SPI slave the duration of the load signal needs to be adjusted. Figure 27 indicates the timing relationship between the clock and load signal. The load goes high during the negative cycle of a positive edge triggered clock ensuring that the throughput of the SPI system is maintained since data can be transferred on every positive edge. This is because the load operation is happening in the negative cycle of the clock and is not wasting any useful clock cycles for that.

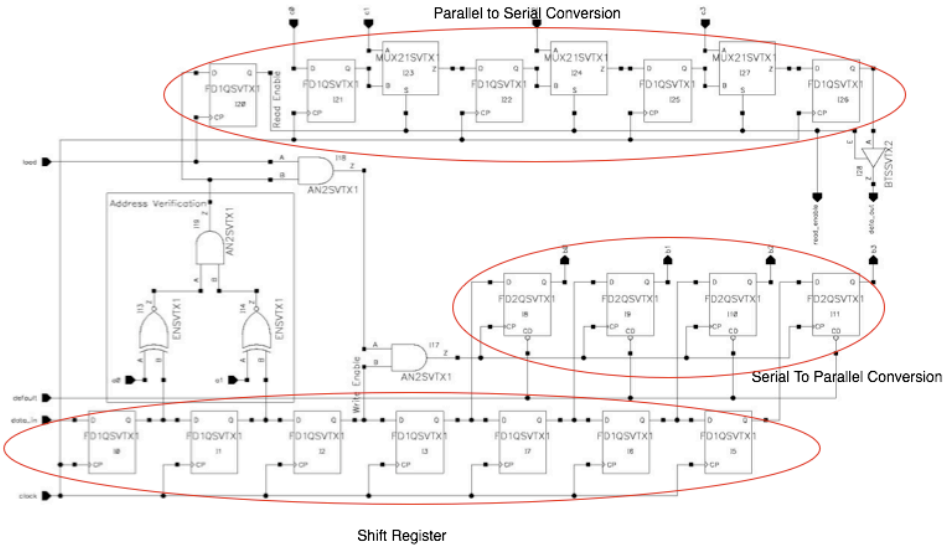

Figure 26: Schematic level description of some functional modules in the SPI

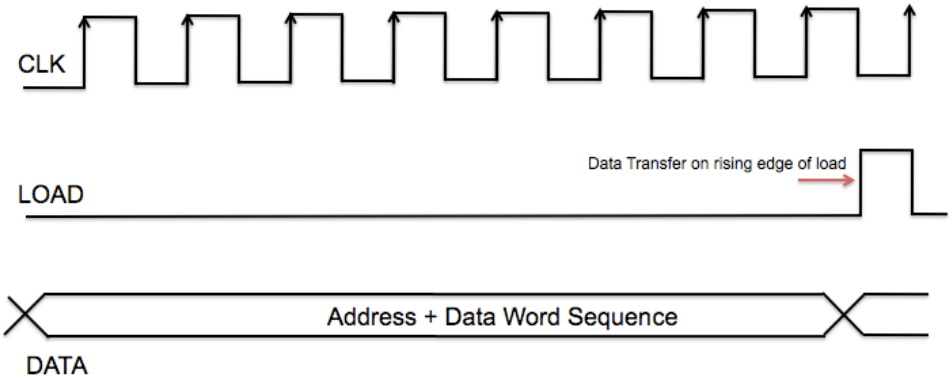

Figure 27: Timing Relationship of clock and load signals

The disadvantage of this approach is that some data bits have to be reserved for loading the address and control bits. The associated logic to process the control sequence will marginally add to the overall area. Even to read from the SPI the master has to first transfer the address packet through the data line. However this is very small compared to the entire data packet size and therefore does not affect the speed of operation. Care must be taken to prevent two SPI slaves from contending for the data out line. This can cause errors in the output received. To ensure correct operation of the read operation a single bit preamble is used to detect the start of a valid sequence. The read enable line shown in Figure 25 is used to keep the data out line at zero when the device is not in read mode.

To prevent contention for the read out line, read and write operations can be performed alternately.

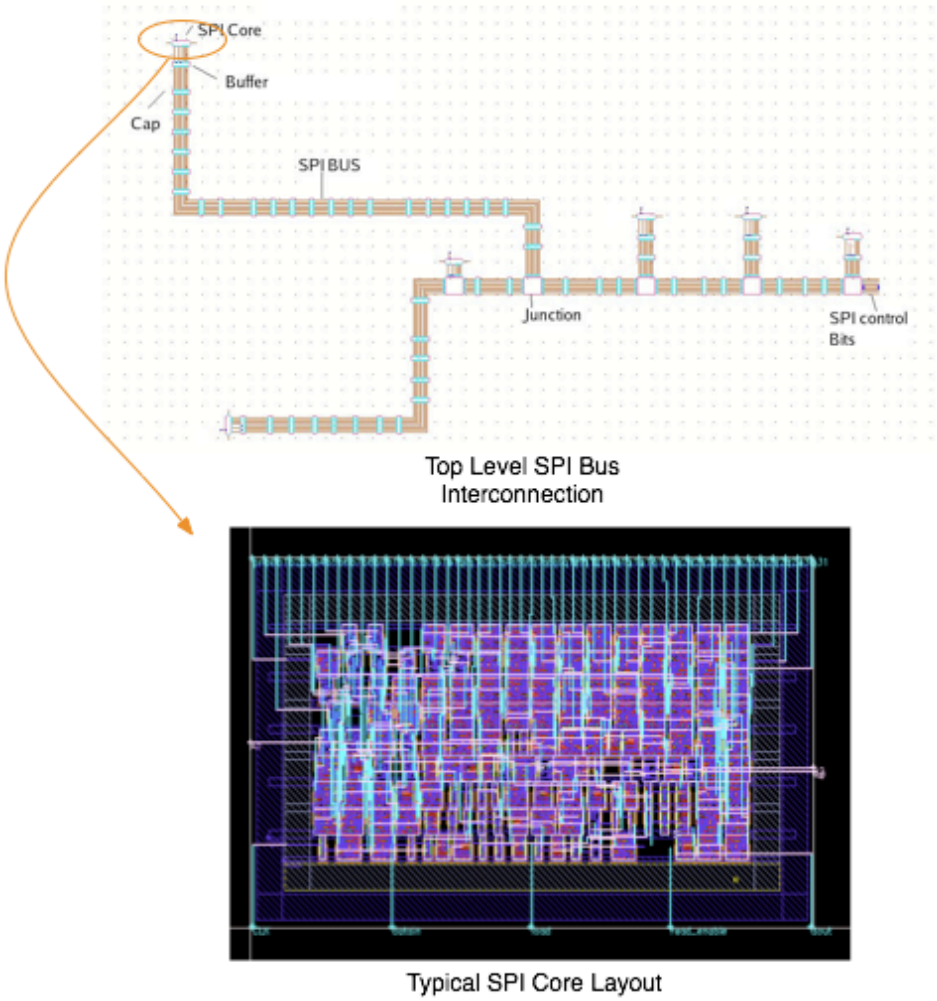

Figure 28: Top level SPI Bus Interconnection

Figure 28 shows the top level SPI bus interconnection scheme used for the address based SPI. Since the SPI slaves can be situated at large distances from each other, the SPI bus needs to be adequately buffered as shown.

CHAPTER VI

CONCLUSION

6.1 List of Important Features and Innovations

- A technique to implement multi-gigabit per second pulse shaping filters is discussed. The technique requires low power and low area.

- Upsampling is implemented by having parallel paths that are multiplexed using the CLK signal at the output (Figure 5). Though the upsampling illustrated here is only for a factor of 2, the same concept can be extended to higher factors by using multi-phase clocks.

- In order to run the FIR at high speeds, all the possible outcomes are precalculated in the digital look-up table based filter implementation. These outcomes can be stored at the addresses representing the last N data inputs corresponding to each outcome.

- If the use of memory is to be avoided, a combinational logic block can be used to implement the address word to bit function for each bit of the output (Figure 5) in each of the parallel paths.

- The even and odd paths have D flip-flops at the end to latch the output. The clock applied to these flip-flops is delayed to compensate for the delay in the combinational blocks.

- The final parallel to serial conversion implementing upsampling can be done for each output bit using a multiplexer (Figure 5). The clock fed to the select terminal of the multiplexers is further delayed to compensate for the clock to output delay of the flip-flops at the output of odd and even paths. The output word then drives the digital-to-analog converter to provide the correct analog output.

- A unique address based SPI has been proposed which uses a single slave select line

and can run faster than the traditional daisy chain SPI configuration.

6.2 Scope for Future Work

The upsampling factor of the filter can be increased from 2 to higher orders by using polyphase clocks. The SPI system's scope can be extended to include features like auto initialization, error detection and correction. This can be achieved by adding some processing power to the SPI master and intelligently using the SPI core to obtain feedback from various slaves and control them based on the feedback.

REFERENCES

[1] A T Erdogan,T Arslan, "Low Power FIR Implementations on Based on Coefficient Ordering Algorithm." VLSI, 2004. Proceedings. IEEE Computer society Annual Symposium on 19-20 Feb. 2004 Page(s):226 - 228.

[2] Cadence Design Systems, "Build Gates User Guide." http://www.cadence.com, (Date:04/2008).

[3] Cadence Design Systems, "SOC Encounter User Guide." http://www.cadence.com,(Date:04/2008).

[4] David Kalinsky,Roee Kalinsky, "Introduction to Serial Peripheral Interface." http://www.embedded.com,(Date:10/2008).

[5] John G Proakis, "Digital Communications." Mc Graw Hill Higher Education.

[6] Ken Gentile, "The care and feeding of digital pulse shaping filters." http://www.rfdesign.com,(Date:04/2008).

[7] Mohamed Almahdi Eshtawie,Masuri Othman, "Implementation of an Optimized Coefficients Pulse Shaping FIR Filters." ICSE2006 Proc. 2006, Kuala Lumpur, Malaysia.

[8] National Instruments Tutorials, "Pulse Shaping Filters in Communication Systems." http://zone.ni.com/devzone/cda/tut/p/id/3876,(Date:10/2008).

[9] National Instruments Tutorials, "Pulse Shaping to Improve Spectral Efficiency." http://zone.ni.com/devzone/cda/ph/p/id/200,(Date:10/2008).

[10] Sangyun Hwang,Gunhee Han, Sungho Kang,Jaeseok Kim, "New Distributed Arithmetic Algorithm for Low Power FIR Implementation." Signal Processing Letters, IEEE Volume 11, Issue 5, May 2004 Page(s):463 - 466 .

[11] Wikipedia Information Resource, "Serial Peripheral Interface Bus." http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus,(Date:10/2008).